擬凸関数を制約に持つDC計画問題に対する外部近似法の改善 (非線形解析学と凸解析学の研究)

AUTHOR(S):

徳重, 友輔; 山田, 修司; 谷野, 哲三

CITATION:

徳重, 友輔 ...[et al]. 擬凸関数を制約に持つDC計画問題に対する外部近似法の改善 (非線形解析学と凸解析学の研究). 数理解析研究所講究録 2010, 1685: 249-258

ISSUE DATE:

2010-04

URL:

http://hdl.handle.net/2433/141437

数理解析研究所講究録
第1685巻 2010年 249-258

擬凸関数を制約に持つDC計画問題に対する外部近似法の改善

(Improvement of an outer approximation method for solving a DC programming problem with pseudo-convex constraint functions)

新潟大学大学院自然科学研究科　徳重 友輔　TOKUSHIGE Yusuke
Graduate School of Science and Technology, Niigata University
新潟大学大学院自然科学研究科　山田 修司　YAMADA Syuuji
Graduate School of Science and Technology, Niigata University
大阪大学大学院工学研究科　谷野 哲三　TANINO Tetsuzo
Graduate School of Engineering, Osaka University

1　はじめに

本研究では, 擬凸関数を制約に持つDC計画問題 (DC programming problem) に対する外部近似法の改善を提案する. DC計画問題に対しては, Tuy [4] により外部近似法に基づく凸多面体近似アルゴリズムが提案されている. しかしながら, Tuyのアルゴリズムに基づく凸多面体近似は対象とする集合が変化することにより効率的でないことが知られている. このため, 本研究では凸多面体近似の効率性を向上させた新たなアルゴリズムを提案する. また, アルゴリズムの各反復において凸多面体を更新する際, 従来は極集合の性質を用いて計算されていた. しかしながら, この手法は各反復において多くの連立一次方程式を解かなければならないため, 計算時間の増加を招いてしまう. このため, 本研究では新たな頂点の計算法を導入する. 導入する手法は, 既存の頂点間の連結情報と各頂点の含まれるファセットの情報を利用する. 各反復において, 既存の頂点間の連結情報を基に新たに生成される頂点を計算し, 各頂点の含まれるファセットの情報を基に新しく生成された頂点間の連結情報を生成する. 本研究では, DC計画問題に対して, 修正アルゴリズムを提案し, 新たな頂点計算方法を導入する. また, 提案する外部近似法の有効性を計算機実験により検証する.

2　DC計画問題

本研究では, 次の数理計画問題について考える.

$$\begin{cases} \text{minimize} & g(\mathbf{x}) - h(\mathbf{x}) \\ \text{subject to} & a_j(\mathbf{x}) \leq 0,\ j = 1, \ldots, m. \end{cases} \tag{1}$$

ただし, $g, h : R^n \to R$ は凸関数, $a_j : R^n \to R$ $(j = 1, \ldots, m)$ は擬凸関数とする. ここで $X := \{\mathbf{x} \in R^n : a_j(\mathbf{x}) \leq 0,\ j = 1, \ldots, m\}$ とする. 問題 (1) の目的関数はDC関数 (difference of two convex functions) であり, 制約集合 X は凸集合であるので, 問題 (1) はDC計画問題となる. さらに問題 (1) に対して, 次を仮定する.

(A1) $\text{int}\, X = \{\, \mathbf{x} \in R^n : a_j(\mathbf{x}) < 0,\ j = 1, \ldots, m\} \neq \phi$. ただし, $\text{int}\, X$ は X の内部集合を表す.

(A2) X はコンパクトである.

(A3) $r \geq \max\{\|\mathbf{x}-\mathbf{y}\| : \mathbf{x},\mathbf{y} \in X\}$ を満たす実数 $r > 0$ が与えられている.

目的関数は連続であり, 仮定 (A2) より制約集合はコンパクトなので, 問題 (1) は最適解を持つことがわかる.

3 Tuy のアルゴリズム

本章では [4] で提案されている Tuy の外部近似アルゴリズムを紹介する.

Tuy の外部近似アルゴリズム.

ステップ 0.

問題 (1) に対して適当な実行可能解 $\mathbf{y}_0 \in X$ を選び, $\omega_0 := g(\mathbf{y}_0) - h(\mathbf{y}_0)$ とする. $\tilde{t} \in R \bigcup \{+\infty\}$, $D_0 := \{(\mathbf{x},t) : \mathbf{x} \in X,\ t \leq \tilde{t},\ g(\mathbf{x}) - t \leq \omega_0\}$ とし, D_0 を含む凸多面体 P_0 を生成する. P_0 の頂点集合 $V(P_0)$ を計算する. また, $\bar{\mathbf{y}} \in \operatorname{int} X$, $\bar{t} < \tilde{t}$, $g(\bar{\mathbf{y}}) - \bar{t} < \bar{\omega}$, $h(\bar{\mathbf{y}}) - \bar{t} < 0$ を満たす $(\bar{\mathbf{y}}, \bar{t})$ をとる. ただし, $\bar{\omega}$ は問題 (1) の最適解の下界の 1 つとする. 反復回数 $k = 1$ としてステップ 1 へ.

ステップ 1. $(\mathbf{x}_k, t_k) \in \operatorname{argmin}\{-h(\mathbf{x}) + t : (\mathbf{x},t) \in V(P_k)\}$ を計算する.

ステップ 1–1. $-h(\mathbf{x}_k) + t_k \geq 0$ ならば, アルゴリズムを停止する. そのとき, $\mathbf{y}_k$ が問題 (1) を解く.

ステップ 1–2. $-h(\mathbf{x}_k) + t_k < 0$ ならば, $\omega_{\mathbf{x}_k} := g(\mathbf{x}_k) - h(\mathbf{x}_k)$ とする.

ステップ 1–2–1. $\omega_k \leq \omega_{\mathbf{x}_k}$ ならば, $\mathbf{y}_{k+1} := \mathbf{y}_k$, $D_{k+1} := D_k$ とする.

ステップ 1–2–2. $\omega_k > \omega_{\mathbf{x}_k}$ ならば, $\mathbf{y}_{k+1} := \mathbf{x}_k$, $D_{k+1} := \{(\mathbf{x},t) : \mathbf{x} \in X,\ t \leq \tilde{t},\ g(\mathbf{x}) - t \leq \omega_{\mathbf{x}_k}\}$ とする.

$\omega_{k+1} := g(\mathbf{y}_{k+1}) - h(\mathbf{y}_{k+1})$ として, ステップ 2 へ.

ステップ 2.

$l_k(\mathbf{x}_k, t_k) > 0$,
$l_k(\mathbf{x},t) \leq 0 \qquad \forall (\mathbf{x},t) \in \{(\mathbf{x},t) : \mathbf{x} \in X,\ t \leq \tilde{t},\ g(\mathbf{x}) - t \leq \omega_{k+1}\}$
を満たすアフィン関数 $l_k(\mathbf{x},t)$ を生成する. ステップ 3 へ.

ステップ 3.

凸多面集合 $P_{k+1} := P_k \bigcap \{(\mathbf{x},t) : l_k(\mathbf{x},t) \leq 0\}$ とし, P_{k+1} の頂点集合 $V(P_{k+1})$ を計算する. ステップ 4 へ.

ステップ 4.

反復回数 $k \leftarrow k+1$ として, ステップ 1 へ.

このアルゴリズムは有限回の反復で問題 (1) の大域的最適解を求める. すべての集積点が大域的最適解となる無限暫定解列 $\{(\mathbf{x}_k, t_k)\}$ を生成する.

Tuy のアルゴリズムの大域的収束性は, 凸集合列 $\{D_k\}$ を凸多面体列 $\{P_k\}$ で近似することで保証される. しかしながら $\{P_k\}$ の近似対象集合が度々変化するため, 近似率の減少を招いてしまう. このため, 次章ではこの点を改善する新たなアルゴリズムを提案する.

4 Tuyのアルゴリズムの改善

本章では, 問題 (1) に対し, Tuy のアルゴリズムを修正した次の逐次近似解法を提案する.

修正アルゴリズム.

ステップ 0.

適当な実行可能解 $\mathbf{y}_0 \in X$ を選び, $\omega_0 := g(\mathbf{y}_0) - h(\mathbf{y}_0)$ とする. また, $h_0(\mathbf{x}) := h(\mathbf{x}) + \omega_0$ とする. $\tilde{t} \in R \bigcup \{+\infty\}$, $D := \{(\mathbf{x}, t) : \mathbf{x} \in X,\ g(\mathbf{x}) \leq t \leq \tilde{t}\}$ とし, D を含む凸多面体 P_0 を生成する. P_0 の頂点集合 $V(P_0)$ を計算する. また, $\bar{\mathbf{y}} \in \operatorname{int} X$, $g(\bar{\mathbf{y}}) < \bar{t} < \tilde{t}$ を満たす $(\bar{\mathbf{y}}, \bar{t})$ をとる. 反復回数 $k = 1$ としてステップ 1 へ.

ステップ 1.

$(\mathbf{x}_k, t_k) \in \operatorname{argmin}\{-h_k(\mathbf{x}) + t : (\mathbf{x}, t) \in V(P_k)\}$ を計算する.

ステップ 1–1. $-h_k(\mathbf{x}_k) + t_k \geq 0$ ならば, アルゴリズムを停止する. このとき, $\mathbf{y}_k$ が問題 (1) を解く.

ステップ 1–2. $-h_k(\mathbf{x}_k) + t_k < 0$ ならば, $\omega_{\mathbf{x}_k} := g(\mathbf{x}_k) - h_k(\mathbf{x}_k)$ とする.

ステップ 1–2–1. $\omega_{\mathbf{x}_k} < 0$ かつ $a_j(\mathbf{x}_k) \leq 0$ $(j = 1, \ldots, m)$ ならば, $h_{k+1}(\mathbf{x}) := h_k(\mathbf{x}) + \omega_{\mathbf{x}_k}$, $\mathbf{y}_{k+1} := \mathbf{x}_k$ とする.

ステップ 1–2–2. $\omega_{\mathbf{x}_k} \geq 0$ または $\max\{a_j(\mathbf{x}_k) : j = 1, \ldots, m\} > 0$ ならば, $h_{k+1}(\mathbf{x}) := h_k(\mathbf{x})$, $\mathbf{y}_{k+1} := \mathbf{y}_k$ とする.

ステップ 2 へ.

ステップ 2.

次を満たすアフィン関数 $l_k(\mathbf{x}, t)$ を生成する.

$l_k(\mathbf{x}_k, t_k) > 0$,

$l_k(\mathbf{x}, t) \leq 0 \qquad \forall (\mathbf{x}, t) \in \{(\mathbf{x}, t) : g(\mathbf{x}) \leq t \leq \tilde{t}, \mathbf{x} \in X\}$.

ステップ 3 へ.

ステップ 3.

凸多面集合 $P_{k+1} := P_k \bigcap \{(\mathbf{x}, t) : l_k(\mathbf{x}, t) \leq 0\}$ とし, P_{k+1} の頂点集合 $V(P_{k+1})$ を計算する. $k \leftarrow k + 1$ として, ステップ 1 へ戻る.

この修正アルゴリズムでは, 凸多面体近似をする対象の集合 D が変化しないことに注意する.

定理 1. (Kubo, Tamura, Matsui [6])
凹最小化問題で, 制約集合が空でない凸多面体ならば, 最適解は制約集合の頂点集合上に存在する.

ステップ 2 の $l_k(\mathbf{x}, t)$ の生成に対して,

$$\beta(\mathbf{x}, t) := \max\{a_1(\mathbf{x}), \ldots, a_m(\mathbf{x}),\ g(\mathbf{x}) - t\}$$

とする. もし, $g(\mathbf{x}_k) - t_k \geq 0$ ならば,

$$\beta(\mathbf{x}_k, t_k) \geq 0$$

となる. また, $g(\mathbf{x}_k) - t_k < 0$, $a_j(\mathbf{x}_k) \leq 0$ $(j = 1, \ldots, m)$ ならば,

$$-h_k(\mathbf{x}_k) + g(\mathbf{x}_k) < -h_k(\mathbf{x}_k) + t_k$$

となる. $\mathbf{x}_k \in X$ より, $(\mathbf{x}_k, g(\mathbf{x}_k)) \in D \subset P_k$ となるので, 定理1に矛盾する. よって, $g(\mathbf{x}_k) - t_k < 0$ ならば, $a_{j'}(\mathbf{x}_k) > 0$ を満たす $j' \in \{1, \ldots, m\}$ が存在する. したがって,

$$\beta(\mathbf{x}_k, t_k) > 0$$

となる. 一方,

$$\beta(\bar{\mathbf{y}}, \bar{t}) < 0$$

である. よって, $(\mathbf{x}_k, t_k)$ と $(\bar{\mathbf{y}}, \bar{t})$ は超平面 $\{(\mathbf{x}, t) : \beta(\mathbf{x}, t) = 0\}$ で強分離される. そこで, $(\mathbf{x}_k, t_k)$ と $(\bar{\mathbf{y}}, \bar{t})$ を結んだ線分と, $\{(\mathbf{x}, t) : \beta(\mathbf{x}, t) = 0\}$ の交点を (ξ_k, θ_k) とし, $s_k \in \partial\beta(\xi_k, \theta_k)$ を選び,

$$l_k(\mathbf{x}, t) = \langle s_k,\ (\mathbf{x}, t) - (\xi_k, \theta_k) \rangle$$

と定義する. ただし, $\partial\beta(\xi_k, \theta_k)$ は (ξ_k, θ_k) における β の劣微分を表し, $\langle \cdot, \cdot \rangle$ は R^n における内積を表す.

次に修正アルゴリズムが収束することを証明する.

定理 2.
修正アルゴリズムで生成される暫定解列 $\{(\mathbf{x}_k, t_k)\}$ の任意の集積点 $(\bar{\mathbf{x}}, \bar{t})$ は D に含まれる.

定理 3.
修正アルゴリズムで生成される暫定解列 $\{(\mathbf{x}_k, t_k)\}$ に対して次が成立する.

$$-h_k(\mathbf{x}_k) + t_k \to 0 \quad \text{as} \quad k \to \infty.$$

この2つの定理より, 暫定解列 $\{\mathbf{y}_k\}$ の任意の集積点は問題 (1) の大域的最適解になる.

5 頂点間の連結関係を用いた外部近似法

修正アルゴリズムのステップ3における P_{k+1} の更新に伴い, P_{k+1} の頂点集合 $V(P_{k+1})$ を求める必要がある.

従来, 頂点集合 $V(P_{k+1})$ の計算手法として, 極集合の性質を用いた方法が用いられていた. しかし, この手法では各反復において多くの連立一次方程式を解くことになるため, 計算時間が膨大になることが知られている. そこで, 本論文では頂点間の連結関係を用いた頂点集合の計算手法を提案する. このため, 次を定義する.

定義 1. [**凸包, アフィン包**] S を R^n の部分集合とする. このとき, S の凸包 co$S$ は

$$\text{co } S := \{u \in R^n : u = \lambda_1 u^1 + \lambda_2 u^2,\ \lambda_1 + \lambda_2 = 1,\ \lambda_1, \lambda_2 \geq 0,\ u^1, u^2 \in S\}.$$

で表される. また, S のアフィン包 aff$S$ は

$$\text{aff } S := \{u \in R^n : u = \lambda_1 u^1 + \lambda_2 u^2,\ \lambda_1 + \lambda_2 = 1,\ \lambda_1, \lambda_2 \in R,\ u^1, u^2 \in S\}.$$

で表される.

定義 2. [面, ファセット, 辺] T を R^n の空でない凸多面集合とする. T の空でない部分集合 F が次の条件をすべて満たすとき, F は T の面と呼ばれる.

1. $F = (\text{aff } F) \cap T$.
2. $(\text{co } \{u^1, u^2\}) \cap F = \emptyset \quad \forall u^1, u^2 \in T \backslash F$.

ただし, 便宜上 T 自身も T の面であるとする. また, F が T の面であり, F の次元数が $n-1$ のとき, F は T のファセットと呼ばれる. さらに, F が T の面であり, F の次元数が 1 のとき, F は T の辺と呼ばれる.

定義 3. [頂点の連結] T を R^n の空でない凸多面集合とし, $v', v''(v' \neq v'')$ を T の頂点とする. このとき, co $\{v', v''\}$ が T の辺ならば v', v'' は連結しているという.

5.1 初期段階における頂点集合の計算

修正アルゴリズムのステップ 0 において, 制約集合 X を覆う n 単体 S_n は次のように生成できる.

$$S_n := \text{co}\{V_1, \ldots, V_{n+1}\},$$

ただし, $V_{n+1} := \mathbf{y}_0 + \sum_{i=1}^{n} re^i$, $V_i := V_{n+1} - r(1+\sqrt{n})e^i$. さらに D を覆う凸多面対 P_0 は次にしたがって生成できる.

$$P_n := \text{co}\left\{ \begin{pmatrix} V_1 \\ \alpha+\beta \end{pmatrix}, \ldots, \begin{pmatrix} V_{n+1} \\ \alpha+\beta \end{pmatrix}, \begin{pmatrix} V_1 \\ \gamma-\beta \end{pmatrix}, \ldots, \begin{pmatrix} V_{n+1} \\ \gamma-\beta \end{pmatrix} \right\},$$

ただし, $\alpha := \max\{g(\mathbf{x}) : \mathbf{x} \in S_n\}$, $\beta := \max\{h(\mathbf{x}) : \mathbf{x} \in S_n\}$, $\gamma := \min\{\langle \mathbf{p}, \mathbf{x} - \mathbf{y}_0 \rangle + g(\mathbf{y}_0) : \mathbf{x} \in S_n\}$ ($\mathbf{p} \in \partial g(\mathbf{y}_0)$). このとき, 頂点の生成の方法からファセットの数は $n+3$ である. 任意の $i \in \{1, \ldots, 2(n+1)\}$ に対して, $v(i)$ に連結する頂点の添え字集合 $\mathcal{V}_i$ と $v(i)$ を含むファセットの添え字集合 $\mathcal{F}_i$ は次で与えられる.

$$\mathcal{V}_i := \begin{cases} (\{1, \ldots, n+1\} \backslash \{i\}) \cup \{n+1+i\} & \text{if } i \in \{1, \ldots, n+1\}, \\ (\{n+2, \ldots, 2(n+1)\} \backslash \{i\}) \cup \{i-(n+1)\} & \text{if } i \in \{n+2, \ldots, 2(n+1)\}, \end{cases}$$

$$\mathcal{F}_i := \begin{cases} \{1\} \cup \{2, \ldots, n+2\} \backslash \{i+1\} & \text{if } i \in \{1, \ldots, n+1\}, \\ \{2, \ldots, n+2\} \backslash \{i+1-(n+1)\} \cup \{n+3\} & \text{if } i \in \{n+2, \ldots, 2(n+1)\}. \end{cases}$$

5.2 切除される頂点の計算

修正アルゴリズムの反復 k において, 任意の $i \in \{1, \ldots, \ell\}$ に対して, $v(i) = (\mathbf{x}_i, t_i)$ と連結している頂点の添え字集合 $\mathcal{V}_i$ と $v(i)$ を含むファセットの添え字集合 $\mathcal{F}_i$ が与えられているものとする. すなわち, すべての $i \in \{1, \ldots, \ell\}$ に対して次が成立する.

- 任意の $j \in \mathcal{V}_i$ に対して, co $\{v(i), v(j)\}$ は (P_k) の辺である.
- 任意の $j \in \mathcal{F}_i$ に対して, $v(i) \in F_j$ である.

修正アルゴリズムのステップ 2 において, $l_k(\mathbf{x},t)$ の生成方法および P_{k+1} の更新方法から $(\mathbf{x}_k,t_k) \notin V(P_{k+1})$ が成立する. したがって, $(\mathbf{x}_k,t_k)$ と連結している頂点を順次調べることで, 切除される頂点をすべて求めることができる. 切除される頂点をすべて求めるアルゴリズムは次のとおりである.

アルゴリズム A(切除されるすべての頂点の計算)

ステップ 0.

$\mathcal{M} := \mathcal{M}' := \{i_k\}$ とする. ただし, i_k は $v(i_k) = (\mathbf{x}_k,t_k)$ を満たす. ステップ 1 へ.

ステップ 1.

$\mathcal{M}' = \emptyset$ ならば, アルゴリズムを終了する. その他の場合, $j \in \mathcal{M}'$ を選び, ステップ 2 へ.

ステップ 2.

ステップ 2–0. $\mathcal{T} = \mathcal{V}_j$ とし, ステップ 2–1 へ.

ステップ 2–1. $\mathcal{T} = \emptyset$ ならば, ステップ 3 へ. その他の場合, $s \in \mathcal{T}$ を選び, ステップ 2–2 へ.

ステップ 2–2. $l_k(\mathbf{x}_s,t_s) > 0$ かつ $s \notin \mathcal{M}$ ならば, $\mathcal{M} \leftarrow \mathcal{M} \cup \{s\}$, $\mathcal{M}' \leftarrow \mathcal{M}' \cup \{s\}$ とする. ステップ 2–3 へ.

ステップ 2–3. $\mathcal{T} \leftarrow \mathcal{T} \backslash \{s\}$ として, ステップ 2–1 へ戻る.

ステップ 3.

$\mathcal{M}' \leftarrow \mathcal{M}' \backslash \{j\}$ として, ステップ 1 へ戻る.

アルゴリズム A によって生成される $\mathcal{M}$ に対して, 次が成立する.

$$\forall i \in \mathcal{M},\ v(i) \in V(P_k) \text{ and } v(i) \notin V(P_{k+1})$$

5.3 新たに生成される頂点の計算

修正アルゴリズムの反復 k のステップ 2 で新たに生成される P_{k+1} の頂点を $v \in V(P_{k+1}) \backslash V(P_k)$ とすると, 次を満たす $v(i'), v(i'') \in V(P_k)$ が存在する.

$$l_k(\mathbf{x}_{i'},t_{i'}) < 0,\ l_k(\mathbf{x}_{i''},t_{i''}) > 0 \text{ and } v \in \text{co}\ \{v(i'), v(i'')\}.$$

また, 次が成立する.

$$v \in F_j \quad \forall j \in \mathcal{F}_{i'} \cap \mathcal{F}_{i''}.$$

さらに, P_1 のファセットの数は $n+3$ であり, P_{k+1} の定義から, 反復 k で新たに生成される P_{k+1} のファセットは一つであり, F_{n+3+k} とすることができる. このとき, 明らかに $v \in F_{n+3+k}$ が成立する.

アルゴリズム A により, $V(P_k)$ の除かれる頂点の添え字集合 $\mathcal{M}$ が生成されたものとして, 新たに生成される $V(P_{k+1})$ の頂点をすべて求めるアルゴリズムは次のとおりである.

アルゴリズム B(新たに生成されるすべての頂点の計算)

ステップ 0. $\mathcal{W} := \emptyset$, $\mathcal{M}' := \mathcal{M}$, $\ell' := \ell + 1$ として, ステップ 1 へ.

ステップ 1. $\mathcal{M}' = \emptyset$ ならアルゴリズムを停止する. その他の場合, $j \in \mathcal{M}'$ を選び, ステップ 2 へ.

ステップ 2.

ステップ 2–0. $\mathcal{T} = \mathcal{V}_j$ とし, ステップ 2–1 へ.

ステップ 2–1. $\mathcal{T} = \emptyset$ なら, ステップ 3 へ. その他の場合, $s \in \mathcal{T}$ を選び, ステップ 2–2 へ.

ステップ 2–2. $l_k(\mathbf{x}_s, t_s) < 0$ なら, ステップ 2–3 へ. $l_k(\mathbf{x}_s, t_s) = 0$ なら, ステップ 2–4 へ. その他の場合, ステップ 2–5 へ.

ステップ 2–3. $v(\ell')$, $\mathcal{F}_{\ell'}$, $\mathcal{V}_{\ell'}$ を次のように定める.

$$
\begin{aligned}
&v(\ell') := (1-\lambda)v(j) + \lambda v(s), \\
&\qquad \text{ただし } \lambda = \frac{l_k(\mathbf{x}_j, t_j)}{l_k(\mathbf{x}_j, t_j) - l_k(\mathbf{x}_s, t_s)}, \\
&\mathcal{F}_{\ell'} := (\mathcal{F}_j \cap \mathcal{F}_s) \cup \{n+3+k\}, \\
&\mathcal{V}_{\ell'} := \{s\}.
\end{aligned}
$$

また,

$$\mathcal{V}_s \leftarrow (\mathcal{V}_s \backslash \{j\}) \cup \{\ell'\}$$

とする. さらに, $\mathcal{W} \leftarrow \mathcal{W} \cup \{\ell'\}$, $\ell' \leftarrow \ell' + 1$ として, ステップ 2–5 へ.

ステップ 2–4. $\mathcal{F}_s$, $\mathcal{V}_s$ と次のように修正する.

$$
\begin{aligned}
&\mathcal{F}_s \leftarrow \mathcal{F}_s \cup \{n+3+k\}, \\
&\mathcal{V}_s \leftarrow \mathcal{V}_s \backslash \{j\}.
\end{aligned}
$$

また, $\mathcal{W} \leftarrow \mathcal{W} \cup \{s\}$ として, ステップ 2–5 へ.

ステップ 2–5. $\mathcal{T} \leftarrow \mathcal{T} \backslash \{s\}$ として, ステップ 2–1 へ戻る.

ステップ 3. $\mathcal{M}' \leftarrow \mathcal{M}' \backslash \{j\}$ として, ステップ 1 へ戻る.

アルゴリズム B より, $V(P_{k+1})$ は次のように与えられる.

$$V(P_{k+1}) = \{v(i) : i \in \{1, \ldots, \ell' - 1\} \backslash \mathcal{M}\}.$$

また, 任意の $i \in \mathcal{W}$ に対して, $v(i) \in F_{n+3+k}$ が成り立つ.

5.4 頂点間の連結情報の構築

アルゴリズム B より, $V(P_{k+1})$ のすべての頂点に対してファセットに関する情報は構築されており, 任意の $i \in (\{1, \ldots, \ell\} \backslash \mathcal{W}) \backslash \mathcal{M}$ に対する頂点 $v(i)$ の連結情報は与えられている. また, 任意の $i \in \mathcal{M}$ に対する頂点 $v(i)$ の F_{n+3+k} に含まれない頂点との連結情報も与えられている. したがって, 新たに構築すべき頂点間の連結情報は $\{v(i) : i \in \mathcal{W}\}$ 上で構築すればよいことが分かる. ここで, 次の定理が成立する.

定理 4. $v(i'), v(i'')$ を P_{k+1} の頂点とする. このとき, co $\{v(i'), v(i'')\}$ が P_{k+1} の辺であるための必要十分条件は $|\mathcal{F}_{i'} \cap \mathcal{F}_{i''}| = n - 1$ である. ただし, $|\mathcal{F}_{i'} \cap \mathcal{F}_{i''}|$ は $\mathcal{F}_{i'} \cap \mathcal{F}_{i''}$ の元の個数を表す.

ここで, $\mathcal{W}=\{i_1,\ldots,i_t\}$ とすると, 次のアルゴリズムより, $\{v(i):i\in\mathcal{W}\}$ での頂点の連結情報を構築することができる.

アルゴリズム C(頂点間の連結情報の構築)

ステップ 0. $j=1$ として, ステップ 1 へ.

ステップ 1. $j=t$ なら, アルゴリズムを終了する. その他の場合, ステップ 2 へ.

ステップ 2.

ステップ 2–0. $s=j+1$ とし, ステップ 2–1 へ.

ステップ 2–1. $s=t+1$ なら, ステップ 3 へ. その他の場合, ステップ 2–2 へ.

ステップ 2–2. $|\mathcal{F}_{i_j}\cap\mathcal{F}_{i_s}|=n-1$ なら, $\mathcal{V}_{i_j}\leftarrow\mathcal{V}_{i_j}\cup\{i_s\}$, $\mathcal{V}_{i_s}\leftarrow\mathcal{V}_{i_s}\cup\{i_j\}$ とする. ステップ 2–3 へ.

ステップ 2–3. $s\leftarrow s+1$ として, ステップ 2–1 へ戻る.

ステップ 3. $j\leftarrow j+1$ として, ステップ 1 へ戻る.

したがって, 修正アルゴリズムのステップ 0 において, $V(P_1)$ を生成し, 各反復 k のステップ 3 においてアルゴリズム A, B, C を用いることで, $V(P_{k+1})$ および頂点間の連結情報を構築することができる. ただし, 各反復 k でアルゴリズム A, B, C を効率的に実行するためには, ステップ 3 の終了後に $V(P_{k+1})$ に含まれる頂点の添え字の番号を昇順 (あるいは降順) に整理する必要がある.

6 数値実験

修正アルゴリズムの有効性を検証するために, 次の問題に対して数値実験を行う.

$$\begin{cases} \text{minimize} & g(x)-h(x):=\left(\dfrac{1}{2}\displaystyle\sum_{i=1}^{n}p_{ai}x_i^2-\sum_{i=1}^{n}p_{bi}x_i+p_c\right)-\left(\dfrac{1}{2}\displaystyle\sum_{i=1}^{n}q_{ai}x_i^2-\sum_{i=1}^{n}q_{bi}x_i+q_c\right) \\ \text{subject to} & a(x):=\dfrac{1}{2}\displaystyle\sum_{i=1}^{n}a_i(x_i-b_i)^2-c\le 0,\ x\in R^n, \end{cases} \tag{2}$$

ただし, $p_{ai},p_{bi},p_c,q_{ai},q_{bi},q_c,a_i,b_i,c\in\{1,\ldots,10\}$ $(i\in\{1,\ldots,n\})$ とする.
ここで, $g(x)$ と $h(x)$ はいずれも凸関数なので, 目的関数は DC 関数である. また, 制約関数は微分可能な凸関数なので擬凸関数である. 数値実験では, 各次元数 $n=1,\ldots,8$ において, $p_{ai},p_{bi},p_c,q_{ai},q_{bi},q_c,a_i,b_i,c$ の値を乱数を用いて定め, 許容誤差を $\tau=0.001$ として, 20 個の問題に対して修正アルゴリズムを実行した. 実験は新潟大学大学院自然科学研究科の高速演算サーバ (CPU: Xeron MP 3.33GHz–8MB×3, RAM: 6GB, OS: Linux) で行い, プログラミングには C 言語を用いた. 数値実験の結果は表 1, 2 のとおりである. ただし, 表 1, 2 内の V_{last} は修正アルゴリズム が停止したときの凸多面体 P_k の頂点数を表す.

表 1 の計算時間の平均から, 次元数 n が高高 6 以下の問題 (2) に対して, 修正アルゴリズムが有効であることがわかる. また, 問題の次数が n の時, 外部近似は $n+1$ で行われていることに注意する.

表 1: 修正アルゴリズムの数値実験結果 (平均)

n	Number of iteration	CPU-time(sec)	V_{last}
1	3	0	6
2	17	0	38
3	41	0.001	197
4	77.55	0.011	1075.7
5	76.5	0.066	2814.6
6	264.2	21.361	52707.8
7	280.3	373.478	191452.4
8	243.35	4626.178	540451.8

表 2: 修正アルゴリズムの数値実験結果 (標準偏差)

n	Number of iteration	CPU-time(sec)	V_{last}
1	0	0	0
2	0	0	0
3	0	0.003	0
4	6.201	0.002	48.651
5	23.848	0.050	1266.831
6	276.132	62.826	80698.049
7	155.785	634.375	160782.952
8	75.510	4626.178	264155.460

7 おわりに

本研究では, 問題 (1) に対し, Tuy[4] が提案した外部近似法を改善した新たなアルゴリズムを提案した. ここでは頂点間の連結関係を用いた新たな頂点計算方法を導入した. また, 提案したアルゴリズムが大域的収束性を持つことを示した. さらに, 数値実験により修正アルゴリズムの有効性を検証した.

参考文献

[1] P. Hartman, "On Functions Representable as a Difference of Convex Functions, " Pacific Journal of Mathematics, pp.707–713, 1959.

[2] R. T. Rockafellar, "Convex Analysis," Princeton, 1970.

[3] H. Tuy, "Convex Analysis and Global Optimization," Kluwer Academic Publishers, 1998.

[4] H. Tuy, "TECHNICAL NOTE: On Global Optimality Conditions and Cutting Plane Algorithms," Journal of Optimization Theory and Applications, Vol.118, pp.201–216, 2003.

[5] S. Yamada, T. Tanaka and T. Tanino: Outer Approximation Method Incorporating a Quadratic Approximation for a DC Programming Problem, Journal of Optimization Theory and Applications(to appear)

[6] 久保幹雄, 田村明久, 松井知己編, 応用数理計画ハンドブック, 朝倉書店, 2002.